



App PWA com Typescript Angular

Angular PWA

Os aplicativos angulares são um single page web application traduzindo "aplicativo de uma página"; ou seja, quando o estado é alterado, a página não é recarregada. Em vez disso, o Document Object Model (DOM) é modificado para alterar seu conteúdo. Angular implementa isso usando um esquema virtual DOM e um detector de mudança de estado, de modo que quando o estado do aplicativo é alterado, apenas o DOM é modificado para o mínimo necessário para refletir a mudança.

Agora vamos para criação do nosso primeiro projeto.

Para isso é necessário instalar o Node Js o link a seguir contém as instruções para instalação https://nodejs.org/en/

Teremos que instalar o Angular CLI que consiste num conjunto de comandos, bibliotecas e instruções do Angular. O comando abaixo instala o Angular CLI através do NPM do node.

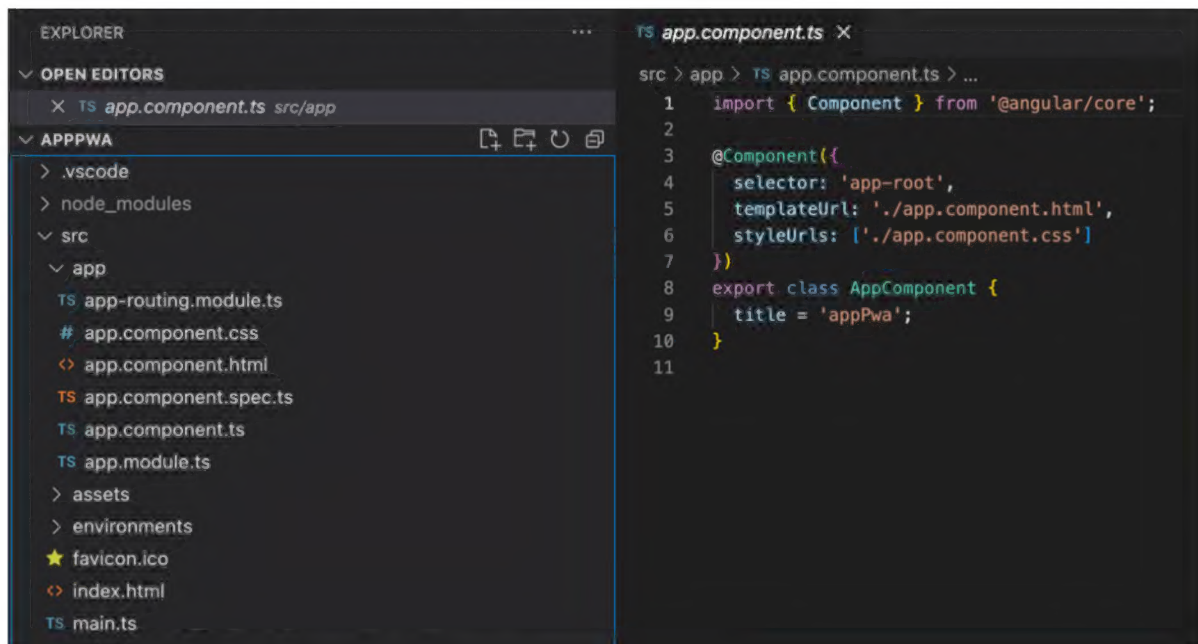
npm install -g @angular/cli

Agora já podemos criar nosso primeiro projeto com o comando abaixo.

ng new descomplica

O Angular CLI nesse momento, uma aplicação Angular básica com apenas um componente.

A imagem abaixo demonstra o projeto criado.



Para entender um pouco sobre a arquitetura de componentes temos que pensar que no Angular sempre temos um componente pai denominado app.components.ts esse componente tem um arquivo HTML um arquivo de module que serve para importar as classes de dependência do projeto. Esse componente pai pode ter vários componentes filhos geralmente se criar um componente de rotas e dentro desse componente renderizamos nossos componentes que vamos denominar páginas.

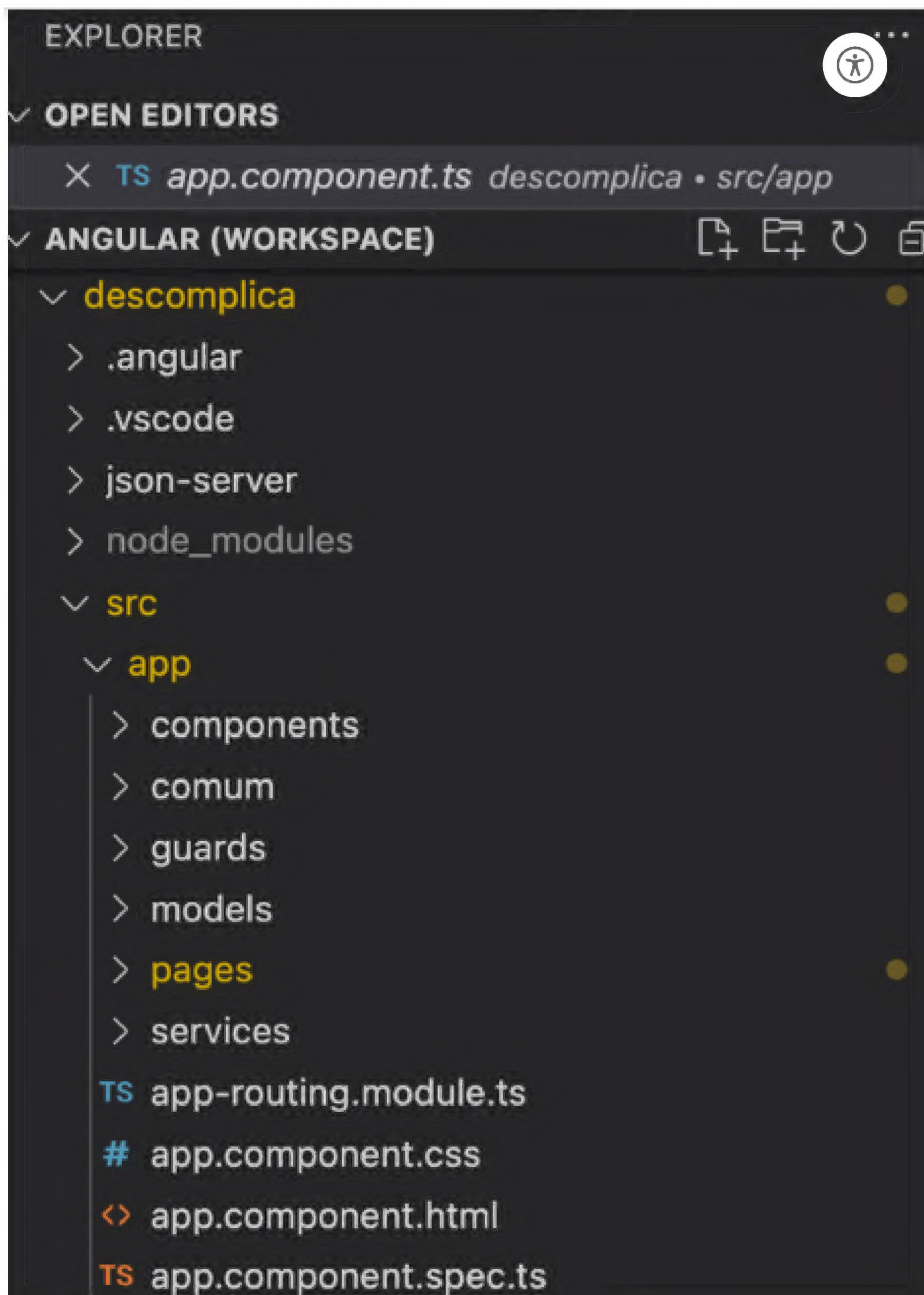
A pasta node_modules contém os códigos das dependências do projeto você pode incluir quantas dependências forem necessários geralmente se usa o NPM para adicionar uma nova dependência.

Nesse projeto vamos utilizar o modelo de rotas abaixo e vamos no decorrer da disciplina adicionar novas rotas no projeto. Uma rota é a

configuração do caminho do componente página que deve ser carregado no componente app-routing.module.ts.

```
const routes: Routes = [
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'cadastro', component: CadastroComponent },
  { path: 'editar', component: EditarComponent },
  { path: 'listar', component: ListarComponent },
  { path: 'listarSimples', component: ListaSimplesComponent },
  { path: 'detalhe/:id', component: DetalheComponent},
  { path: 'json', component: ManipulandoJsonComponent},
  { path: 'parametro', component: ParametroComponent},
  { path: 'modal', component: ModalComponent},
  {
    path: 'privado',
    component: PrivadoComponent,
    canActivate: [AutorizadoGuard]
  },
  {
    path: '**',
    redirectTo: 'home'
  }
];
```

Agora vamos para a configuração das pastas a imagem abaixo demonstra as pastas já criadas. A pasta pages contém os componentes que representam uma página dinâmica no aplicativo, a pasta model contém os objetos e a pasta components contém os componentes filhos que podem ser compartilhados no projeto. Outras pastas serão criadas durante a disciplina.



Agora vamos criar um componente pai e um filho e a partir desses componentes passar informações entre eles. Utilize o comando abaixo para criar os componentes.

ng g componentes pages/home



ng g componentes componentes/header

Esse será o conteúdo do nosso componente home(pai).

```
export class HomeComponent implements OnInit {
  title = 'descomplica';
  constructor() { }
  nome:string = "Professor Bruno Hauck";
  ngOnInit(): void {
  }
}
```

Vamos passar a informação nome “Professor Bruno Hauck” para o nosso componente filho para isso inclua no HTML do componente home o seguinte código.

```
<app-header [nome]="nome" ></app-header>
```

Vamos passar a informação nome “Professor Bruno Hauck” para o nosso componente filho para isso inclua no HTML do componente home o seguinte código.

```
export class HeaderComponent implements OnInit {

  constructor() { }
  @Input() nome:string = '';
  ngOnInit(): void {
  }
}
```

Agora para mostrar a informação no HTML do componente header utilizamos o código abaixo:

```
<app-header [nome]="nome" ></app-header>
```



No decorrer da disciplina vamos adicionar novos códigos para criar um App completo com cadastro, login e listagem dentre outras funcionalidades.

Frameworks de Layout para Angular

O Angular Material é um conjunto de código de Layout que permite criar aplicações responsivas *mobile first* e ajuda muito na construção de sistemas em Angular. Além disso, é baseado no Material Design do *Google*. Você pode encontrar mais informações no site oficial no link abaixo:

Link: https://material.angular.io/components/categories **(Acesso em 04/11/2022)**

O Angular Material fornece uma série de componentes já estilizados e você pode conferir no link indicado acima.

Agora, vamos criar um objeto do tipo Routes para essa configuração, lembrando que vamos adicionar novas rotas no decorrer das aulas.

```
const routes: Routes = [
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'cadastro', component: CadastroComponent },
  { path: 'editar', component: EditarComponent },
  { path: 'listar', component: ListarComponent },
  { path: 'listarSimples', component: ListaSimplesComponent },
  { path: 'detalhe/:id', component: DetalheComponent},
  { path: 'json', component: ManipulandoJsonComponent},
  { path: 'parametro', component: ParametroComponent},
  { path: 'modal', component: ModalComponent},
```

```
  {
   path: 'privado',
   component: PrivadoComponent,
   canActivate: [AutorizadoGuard]
  },
  {
   path: '**',
   redirectTo: 'home'
  }
];
```

Vamos olhar um pouco sobre o ponto de vista mais de layout e construir um formulário simples em HTML, adicionando os componentes do Angular Material.

```
<form [formGroup]="addressForm" novalidate (ngSubmit)="onSubmit()">
 <mat-card class="shipping-card">
  <mat-card-header>
   <mat-card-title>Cadastro</mat-card-title>
  </mat-card-header>
  <mat-card-content>
   <div class="row">
    <div class="col">
     <mat-form-field class="fill">
      <input matInput placeholder="Nome" formControlName="firstName">
      <mat-error *ngIf="addressForm.controls['firstName'].hasError('required')">
       O nome é  <strong>obrigatório</strong>
      </mat-error>
     </mat-form-field>
    </div>
   </div>
   <div class="row">
    <div class="col">
     <mat-form-field >
      <mat-label>Enter your email</mat-label>
      <input matInput placeholder="pat@example.com" formControlName="email" >
      <mat-error *ngIf="addressForm.controls['email'].hasError('required')"></mat-error>
     </mat-form-field>
```

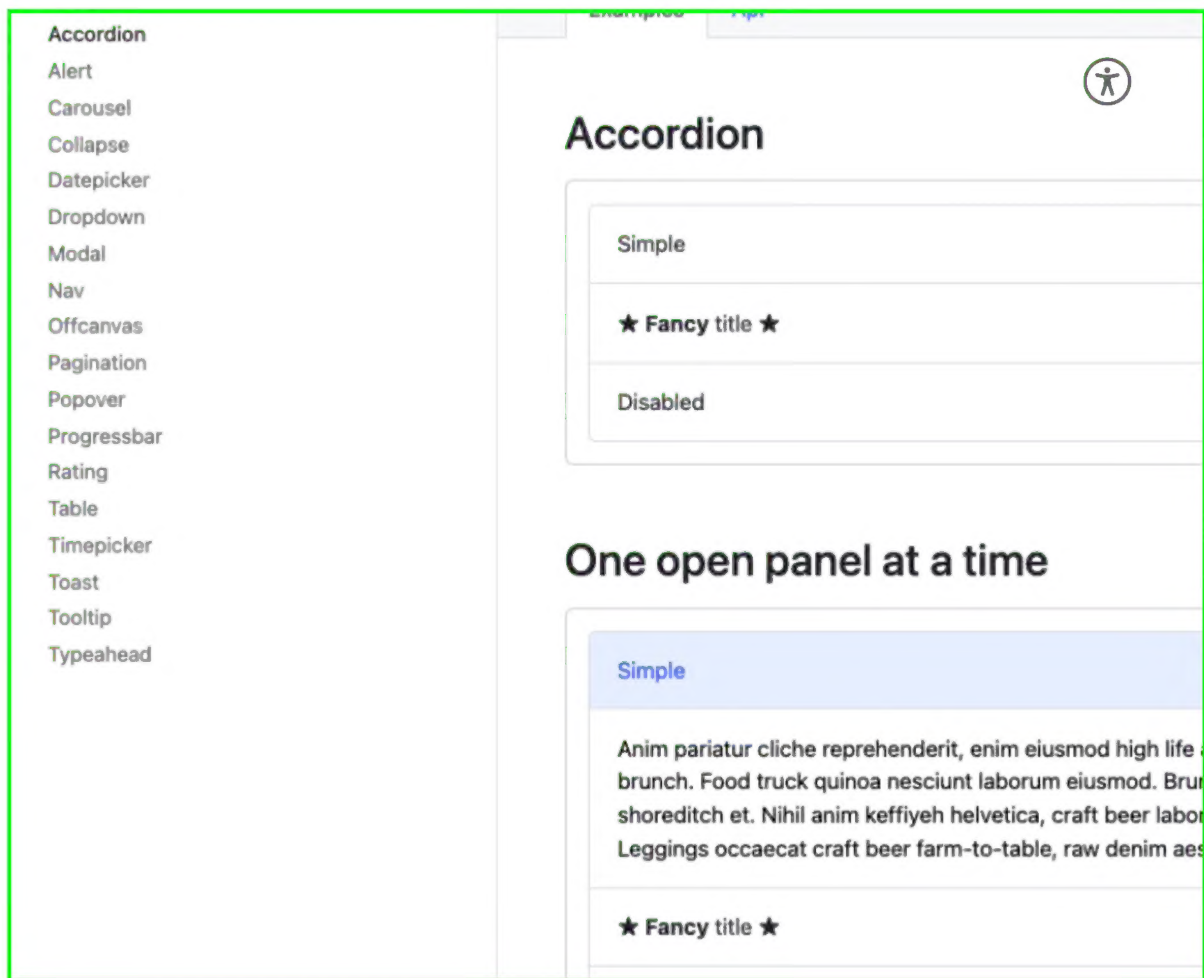
```
    </div>
  </div>
  <div class="row">
    <div class="col">
      <mat-form-field class="fill">
        <input matInput placeholder="Phone" formControlName="phone">
        <mat-error *ngIf="addressForm.controls['phone'].hasError('required')">
          Telefone é <strong>obrigatório</strong>
        </mat-error>
      </mat-form-field>
    </div>
  </div>
  <div class="row">
    <div class="col">
      <mat-form-field class="fill">
                                          <input    matInput    placeholder="Senha"    type="password"
formControlName="password">
        <mat-error *ngIf="addressForm.controls['password'].hasError('required')">
          Senha é <strong>obrigatório</strong>!
        </mat-error>
      </mat-form-field>
    </div>
  </div>
  <div class="row">
    <div class="col">
      <mat-form-field class="fill">
                                      <input    matInput    placeholder="CPF"    mask="000.000.000-00"
formControlName="cpf">
        <mat-error *ngIf="addressForm.controls['cpf'].hasError('required')">
          CPF é  <strong>obrigatório</strong>
        </mat-error>
        <mat-error *ngIf="addressForm.controls['cpf'].getError('cpfNotValid')">
          CPF não é <strong>Valido</strong>
        </mat-error>

      </mat-form-field>
    </div>
```

```
    </div>
  </mat-card-content>
  <mat-card-actions>
          <button mat-raised-button color="primary" [disabled]="!addressForm.valid"
type="submit">Login</button>
  </mat-card-actions>
 </mat-card>
</form>
```

O mais importante é que começamos a construir nosso app usando a arquitetura de componentes, rotas e o Angular Material como layout.

O Boostrap é um framework de layout similar ao Angular Material ele também é outra opção bem popular geralmente utilizado em aplicação PHP para Angular temos uma dependência denominada NG Boostrap que é um conjunto de componentes e pode ser encontrado no link a seguir https://ng-bootstrap.github.io/#/home existem vários tipos de componentes conforme a figura a seguir.



Podemos comparar o formulário Angular Material anterior com código abaixo em Ng Bootstrap

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp"
placeholder="Enter email">
    <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone
else.</small>
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1"
placeholder="Password">
  </div>
  <div class="form-group form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

Porém, durante essa disciplina, vamos utilizar o Angular Mate ara criar nosso projeto base.

Atividade Extra

Além do Angular material existem outros frameworks de Layout ou até mesmo mobile baseados no Angular como Ngx Bootstrap e IONIC que, inclusive, é muito interessante, pois além de um conjunto de componentes é possível gerar aplicações multiplataforma como Web, Android e IOS, sendo o framework mobile baseado em Angular mais popular.

Para complementar seus estudos você pode continuar a usar a aplicação sugerida na aula passada e criar uma arquitetura de rotas e um formulário diferente com, por exemplo, cadastro de produto e uma página home para listar os produtos futuramente. Você pode experimentar outros componentes de Layout, como Ngx Bootstrap e o IONIC.

Referência Bibliográfica

ANGULAR DOCS. **Angular IO**. Disponível em: <https://angular.io>. **(Acesso em 04/11/2022)**

ANGULAR MATERIAL DOCS. **Angular Material**, Disponível em: <https://material.angular.io>. **(Acesso em 04/11/2022)**

IONIC FRAMEWORK. **IONIC.** Disponível em: <>. https://ionicframework.com **(Acesso em 04/11/2022)**

Ir para exercício

